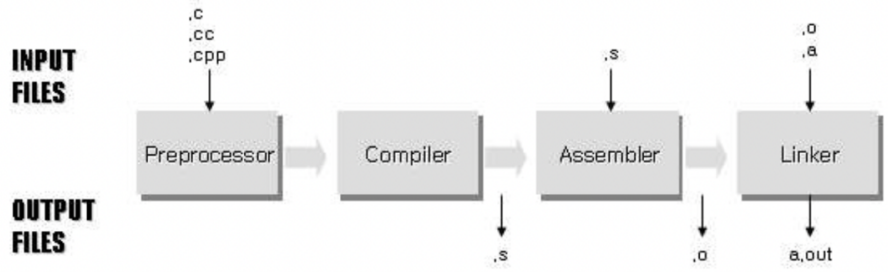
1. 목 적

UNIX 상에서 제공하는 C/C++관련 도구를 미리 사용해 봄으로써, 수업시간 실습이 원활히 진행될 수 있도록 한다.

2. 예비 학습

본문을 읽고 c/c++ 프로그램의 컴파일 과정에 대하여 요약하라. 각 단계별로 하는 일들 과 관련된 도구들 또한 명시하라.

프리프로세싱(preprocessing) - 컴파일링(compiling) -어셈블링(assembling) - 링킹(linking)



1. **프리프로세서(preprocessor) 단계**

: 프리프로세서는 전처리기 또는 선행 처리기라고도 불리며, 소스 코드의 주석을 제거하고 #define 을 치환하는 기능 등을 한다. 코드를 컴파일러가 처리하기 전에 특정 변수를 미리 정의된 문자열로 치환하는 것이다. 전처리기가 실행되면 각 코드 파일에서 지시자(directives)를 찾는데 이 때 지시자는 #으로 시작해서 줄 바꿈으로 끝나는 코드를 말한다.

2. 컴파일러(compiler) 단계

: 컴파일러를 통해 C/C++ 소스파일이 어셈블리 파일로 변환하는 단계다. 대표적인 컴파일러의 예시로 gcc(GNU Compiler Collection), MS Visual C++ compiler등이 있다.
컴파일러의 역할은 크게 프론트엔드(front-end), 미들엔드(middle-end), 백엔드(back-end)로 나눌 수 있다.

1. 프론트엔드 – 소스 코드가 문법에 맞게 작성되었는지 분석한다. 어휘 분석(lexical analysis), 구문 분석(syntax analysis), 의미 분석(semantic analysis), 중간 표현/코드 생성(code generation)을 한다. 어휘 분석에서는 소스 코드를 의미 있는 최소 단위인 토큰(token)으로 나누고, 구문 분석에서 토큰으로 파스 트리(parse tree)를 만들어 문법적 오류를 검출한다. 이 후, 파스 트리로 의미 분석 및 오류 검사를 하고 언어 독립적인 특성을 제공하기 위해 트리 형태의 중간 표현인 Gimple Tree를 생성한다.
2. 미들엔드 – 최적화(optimization)를 수행한다. 프론트엔드에서 받은 Gimple 트리를

SSA(static single assignment) 형태로 변환하고 최적화를 거친 후 최종적으로 백엔드에서 사용할 RTL(register transfer language)를 생성한다.

3. 백엔드 – 역시 최적화를 수행하며 어셈블리 코드를 생성하고 최종적으로 어셈블리어로 구성된 어셈블리 코드 .s 파일이 생성된다.

gcc로 컴파일하는 가장 간단한 방법은 gcc 명령어 뒤에 c 소스 파일을 열거하는 것이다. 예를 들어, gcc main.c 를 하면 파일명 main c 소스코드가 컴파일 된다. 출력되는 .s 파일을 남길 때는 gcc 실행 옵션 중 -s를 하면 된다.

3. 어셈블러(assembler) 단계

: 어셈블리 코드를 0 과 1 로 이루어진 이진 코드인 오브젝트 코드 파일로 변환하는 단계다. 최종적으로 생성되는 .o 파일은 명령어와 자료가 저장된 바이너리 포맷(binary format) 구조를 갖는다. 이 때, 아직 주소 정보는 확정되지 않은 상태이다. 출력되는 오브젝트 파일은 gcc 실행 옵션 -c 를 하면 남길 수 있다.

4. **링커(linker) 단계**

: 어셈블러에 의해 생성된 오브젝트 파일들을 묶어서 실행 코드 파일로 변환한다. 운영체제가 로딩할 수 있도록 주소 정보를 할당한 파일을 생성한다. 프리프로세싱, 컴파일링, 어셈블링을 모두 거쳐 마지막으로 링커에 의해 라이브러리와 링킹 되는 것이다. 이 때 최종적으로 운영체제가 런타임에서 실행할 수 있는 실행 파일이 된다. 별다른 옵션이 없을 경우 출력은 a.out 의 기본 실행 파일으로 되며, -o filename 옵션을 통해 이름 변경이 가능하다.